

# BUNDESREPUBLIK DEUTSCHLAND



## Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

**Aktenzeichen:** 100 38 439.0

**Anmeldetag:** 7. August 2000

**Anmelder/Inhaber:** Siemens AG, München/DE

**Bezeichnung:** Verfahren zum Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungssteuerungen im Kontext der Flow Chart Programmierung

**IPC:** G 05 B, G 06 F

**Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.**

München, den 8. Januar 2001  
**Deutsches Patent- und Markenamt**

Der Präsident  
Im Auftrag

A handwritten signature in black ink, appearing to read "Faust".

Faust

## Beschreibung

Verfahren zum Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungssteuerungen im Kontext der Flow  
5 Chart Programmierung

Die Erfindung bezieht sich auf ein Verfahren für das Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungssteuerungen, wobei ein Anwender mit einem Editor grafische Elemente, insbesondere Kontrollstrukturen und Funktionsblöcke, zu einem auf einer Anzeigeeinrichtung visualisierbaren Flow Chart verknüpft.

Im industriellen Umfeld ist es bekannt, sowohl für die Steuerung eines technischen Prozesses als auch für die Steuerung der Bewegung einer Verarbeitungs- bzw. Produktionsmaschine graphische Eingabehilfsmittel sowie einen Bildschirm zur Visualisierung zu verwenden (Hans D. Kief: "NC/CNC Handbuch", 2000, Hansa Verlag, Seite 254, Bild 7 bzw. Seite 327, Bild 6). Grundelemente grafischer Struktur-, Fluss- und Ablaufpläne sind in der Norm DIN 66 001 aufgeführt.

In "Visuelle Sprachen - ein unaufhaltsamer Trend in der Industrie" (Josef Hübl, SPS/IPC/Drives - Tagungsband, Seite 88 - 95, 23. - 25. November 1999, Nürnberg, Verlag Hüthig GmbH, Heidelberg) ist außerdem angegeben, dass Kontrollfluss- bzw. Datenflussdiagramme für die Steuerung von Automatisierungsaufgaben mit Hilfe graphischer Editoren erstellt werden.

Es ist seit langem üblich in Programmierumgebungen Debugger als Hilfsprogramme für die Fehlersuche und Fehlerlokalisierung einzusetzen (Volker Claus et al., DUDEIN Informatik, 2. erw. Auflage, S. 188, Dudenverlag, 1993).

Bei den heutzutage existierenden Debuggern werden aber die Haltepunkte (Breakpoints), die zum anwenderkontrollierten Ausführen des zu testenden Programmes notwendig sind, in den

Prozessorcode hineincompiliert. Dadurch findet die zum Debuggen notwendige schrittweise bzw. sukzessive Ausführung des Programmes auf einer niedrigen Abstraktionsebene statt und ist dadurch für einen Anwender z.B. hinsichtlich Visualisierung unflexibel.

- Außerdem unterstützen die heutzutage existierenden graphischen Eingabehilfsmittel und graphischen Editoren für die Programmierung von industriellen Steuerungen einen Anwender 10 ungenügend hinsichtlich adaptiver Mechanismen bezüglich seiner Applikation zugrundeliegenden Hardwarekonfiguration und stellen ihm im grafischen Editor nur einen starren und eingeschränkten Vorrat an Sprachmechanismen zur Verfügung.
- 15 Weiterhin unterstützen die heutzutage existierenden graphischen Eingabehilfsmittel und graphischen Editoren für die Programmierung von industriellen Steuerungen nur dediziert die Programmierung der Steuerung eines technischen Prozesses (SPS-Funktionalität) oder die Programmierung der Steuerung 20 der Bewegung einer Verarbeitungs- bzw. Produktionsmaschine. Die Programmerstellung für beide Anwendungsbereiche wird von existierenden Flow Chart-Editoren jeweils nicht adäquat unterstützt.
- 25 Ein weiterer Nachteil der heutzutage bei der Programmierung industrieller Automatisierungsaufgaben eingesetzten Flow Chart-Editoren ist, dass die mit ihnen erzeugten Diagramme direkt in ablaufähigen Prozessorcode umgesetzt werden oder dass aus den Diagrammen ASCII-Code erzeugt wird, der dann im 30 jeweiligen Zielsystem laufzeitintensiv interpretiert werden muss. Neben der dadurch resultierenden Unflexibilität bezüglich der Portierung und Übertragung der Programme auf andere Anlagen oder Maschinen bedeutet dieser Mechanismus als weiteren Nachteil für den Anwender nur eingeschränkte Debugging- 35 Möglichkeiten.

Außerdem liegen zusätzliche Nachteile existierender Flow Chart-Editoren im meist starren und unflexiblen Sprachvorrat an für den Anwender verwendbaren Icons und in der fest vorgegebenen sequentiellen Abarbeitungsreihenfolge der Icons bzw.

- 5 der entsprechenden Funktionsblöcke. Auch bieten existierende Flow Chart-Editoren häufig nur wenig Möglichkeiten zur Formulierung von Synchronisationsmechanismen, die aber insbesondere für die Programmierung von Applikationen in der industriellen Automatisierung sehr oft benötigt werden.

10

Der Erfindung liegt daher die Aufgabe zugrunde, ein Verfahren gemäß Oberbegriff von Anspruch 1 zu schaffen, wobei einem Anwender gemäß der jeweiligen Abstraktionsebene im Programmierstellungsprozess adäquate Debugging-Mechanismen zur Verfügung gestellt werden.

15

Gemäß der Erfindung wird diese Aufgabe für ein Verfahren der eingangs genannten Art dadurch gelöst, dass die folgenden Verfahrensschritte durchgeführt werden:

20

- a) vom Anwender wird ausgehend vom Flow Chart ein Debug-Vorgang vorbereitet,
  - b) dadurch wird jedem grafischen Element ein Suspend-Befehl zugeordnet,
  - c) der Debug-Vorgang wird gestartet,
  - d) der Programmablauf folgt bis zur unmittelbar folgenden Suspend-Befehl,
  - e) dem Anwender wird visualisiert, wo im Flow Chart die aktuelle Flow Chart befindlich ist,
  - f) der Anwender startet das Fortschalten zur nächst möglichen Suspend-Befehl,
  - g) die Schritte d) bis f) werden solange fortgeführt, bis das Flow Chart-Ende erreicht ist.
- 35 Ein Anwender hat dadurch die Möglichkeit das Verhalten bzw. das Fehlverhalten eines Programmablaufs auf grafischer Flow Chart-Ebene zu untersuchen.

Üblicherweise werden die Haltepunkte, die ein Debug-Programm für seine Arbeitsweise benötigt auf Prozessorcodeebene gesetzt, d.h. in den Prozessorcode (durch einen Compiler) eingebracht. In der vorliegenden Erfindung werden aber die Haltepunkte in Form von Suspend-Befehlen den Flow Chart Elementen zugeordnet, d.h. die Einbringung der Haltepunkte geschieht auf Hochsprachenebene.

In der vorliegenden Erfindung kann ein Anwender das Debugging im Single-Step-Modus oder im Breakpoint-Modus durchführen, denn ein Suspend-Befehl kann auch durch Variablenwerte, Bedingungen oder Speicheradressen vorbelegt werden. In Abhängigkeit dieser Vorbelegungen führt sie eine "Suspendierung" (Halt) des Programmablaufs herbei oder nicht.

15

In beiden Modi kann der Anwender das "Durchhangeln" durch das Programm am Bildschirm visuell verfolgen. Das aktuelle grafische Element kann z.B. durch einen Cursor angezeigt werden. Auch andere Möglichkeiten einer Visualisierung sind denkbar.

20

Eine erste vorteilhafte Ausgestaltung der Erfindung liegt darin, dass ein durch einen Suspend-Befehl angehaltenes grafisches Element, bzw. die zu diesem Element gehörige Task mit einem Tasksteuermechanismus des Run-Time-Systems fortgesetzt wird. Der Tasksteuermechanismus kann vom Anwender durch Eingaben im Engineering System bedient werden (im Single-Step-Modus und/oder im Breakpoint-Modus), er kann aber auch durch Programme des Run-Time-Systems angesteuert werden. Z.B. können dadurch Regressionstests durchgeführt werden.

30

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass der Anwender im Engineering System über den Tasksteuermechanismus des Run-Time-Systems einen Resume-Befehl bedient, der den aktuellen Suspend-Befehl fortschaltet. Der Anwender kann durch die einfache Bedienung des Resume-Befehls auf Flow Chart-Ebene sehr einfach ein Einzel-Schritt-Debugging durchführen.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass der Tasksteuermechanismus des Run-Time-Systems über vom Anwender im Engineering System vorbelegbare Variablen in Form eines Breakpoint-Debuggings verwendet wird. Da-

- 5 durch kann der Anwender auf sehr komfortable Art und Weise Breakpoints setzen und auf Flow Chart-Ebene ein Breakpoint-Debugging durchführen.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt  
10 darin, dass die Variablenvorbelegungen im Tasksteuermechanismus durch andere Programme des Run-Time-Systems erfolgen. Dadurch wird das automatische Testen von Programmen (z.B. automatische Regressionstests) sehr leicht ermöglicht.

- 15 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die folgenden Schritte aufeinander folgend durchgeführt werden:

- a) aus dem Flow Chart wird eine textuelle Sprache erzeugt,
- 20 b) die textuelle Sprache wird in einen prozessorunabhängigen Zwischencode kompiliert,
- c) der prozessorunabhängige Zwischencode wird auf die Steuerung geladen,
- 25 d) der prozessorunabhängige Zwischencode wird in ablauffähigen Prozessorcode umgesetzt.

- 30 Dadurch, dass aus den Flow Chart-Diagrammen in einem Zwischen-  
schenschritt eine textuelle Sprache erzeugt wird, hat der Anwender die Möglichkeit, bereits auf dieser Ebene der textuellen Sprache Plausibilitätsüberprüfungen durchzuführen. Er kann aber auch weitere Sprachelemente, die in der textuellen Sprache vorliegen, zu seiner Anwendung hinzubinden. Dadurch,  
35 dass die textuelle Sprache in einem weiteren Zwischen-  
schritt in einen prozessorunabhängigen Zwischencode kompiliert wird,

bleibt die angesprochene Flexibilität für den Anwender weiterhin erhalten. Auch auf dieser Zwischencodeebene kann der Anwender Plausibilitätschecks bzw. ein Debugging durchführen. Der letztendlich in der Steuerung ablaufende Prozessorcode  
5 wird aus dem prozessorunabhängigen Zwischencode generiert, dadurch wird das Target der Anwendung erst zu einem sehr späten Zeitpunkt festgelegt. Durch die Zwischenschritte bei der Codegenerierung können außerdem sehr leicht unterschiedliche Ziel-Hardwaren bedient werden.

10

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass dem Anwender auf Ebene der textuellen Sprache und/oder auf Ebene des Zwischencodes und/oder auf Ebene des Prozessorcodes jeweils eine Debug-Schnittstelle zur Verfügung  
15 steht. Dadurch ist für einen Anwender das Debuggen von Programmen in der jeweiligen Codeebene in der dazugehörigen Abstraktionsstufe möglich. Ein Anwender ist somit in der Lage je nach Ausbildungsstand oder Erfahrung sich eine geeignete Abstraktionsebene zum Debuggen zu wählen. Außerdem ist es bekannt,  
20 dass auf jeder Codeebene bestimmte Typen von Fehlern mehr oder weniger häufig auftauchen. Durch die Möglichkeit des Debuggens auf unterschiedlichen Codeebenen kann ein Anwender gezielt nach Fehlern suchen, die für die jeweilige Codeebene typisch sind. Die Fehler suche bzw. Fehlerlokalisierung wird dadurch effizienter.

25

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass dem Anwender im Flow Chart-Editor, in Abhängigkeit von der zugrundeliegenden Maschinenprojektierung  
30 und/oder Hardwarekonfiguration adäquate Sprachmechanismen zur Verfügung gestellt werden. Dadurch wird einem Anwender eine Programmierumgebung zur Verfügung gestellt, die auf die zugrunde liegende Hardware abgestimmt ist und somit optimal den vorliegenden Anforderungen und Randbedingungen genügt.  
35 Der Sprachvorrat des Flow Chart-Editors adaptiert sich somit selbstständig an die vorhandenen HW-Gegebenheiten (z.B. die zugrunde liegende Maschinenkonfiguration).

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass aus anwenderdefinierten Unterprogrammen einer textuellen Sprache automatisch über einen Umsetzer nach Art eines Compilers weitere grafische Elemente in Flow Chart-Notation generiert werden, welche die Funktionsschnittstelle der entsprechenden Unterprogramme enthalten und die dem Anwender ebenfalls zur Verfügung gestellt werden. Dadurch ist es möglich, dass aus schon vorhandenen Unterprogrammen der textuellen Sprache oder aus zusätzlichen Unterprogrammen, die in die textuelle Sprache eventuell vom Maschinenbauer eingebracht wurden, automatisch Icons und die dazugehörigen Masken vom System generiert werden und dem Anwender im Flow Chart-Editor zur Verfügung gestellt werden. Die Funktionsschnittstelle und die Übergabeparameter der Unterprogramme der textuellen Sprache werden dabei automatisch für die Flow Chart Icons generiert. Durch diesen Mechanismus lassen sich leicht von OEM-Kunden (Original Equipment Manufacturer) schon in textueller Sprache vorliegende Unterprogramme in den Flow Chart-Editor übernehmen. Damit wird dem Endanwender für seine Flow Chart-Programmierung ein angepasster und erweiterter Sprachvorrat an Icons zur Verfügung gestellt.

Für den Hersteller bzw. für den Vertreiber von Flow Chart Editoren für die Programmierung von industriellen Steuerungen ergibt sich weiterhin der Vorteil, dass sie den Flow Chart Editor mit einem Basisvorrat an grafischen Sprachelementen ausliefern können, der dann in Abhängigkeit evtl. schon vorhandener Unterprogramme der textuellen Sprache auf die Belange eines Anwenders angepasst wird. Ein Flow Chart Editor kann somit in einer anpassbaren Standard- oder Basisversion an Kunden ausgeliefert werden (economies of scale). Für den Anwender ergibt sich dadurch die Möglichkeit einer technologischen Skalierung für seine jeweiligen Anwendungen bezüglich des ihm zur Verfügung stehenden Sprachvorrats an grafischen Elementen.

- Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die automatisch generierten grafischen Elemente vom Anwender als Sprachelemente des Flow Charts verwendet werden. Dadurch, dass der Anwender die automatisch generierten Icons als normale Sprachelemente des Flow Chart-Editors verwenden kann, wird der ihm zur Verfügung stehende Sprachvorrat an Flow Chart-Elementen, d.h. an Icons, erweitert. So mit wird die Flexibilität und Ausdrucksmöglichkeit bezüglich der Programmierung von Applikationen für den Anwender erhöht.
- 10 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass als textuelle Sprache Structured Text nach IEC 6-1131 verwendet wird. Dadurch, dass mit IEC 6-1131 eine genormte Sprache auf der Ebene der textuellen Sprache verwendet wird, ist der Austausch bzw. die Kopplung mit anderen Programmiersystemen sehr leicht möglich. Außerdem wird durch die Verwendung von IEC 6-1131 als Zwischensprache die Portierung auf unterschiedliche Zielsysteme sehr erleichtert.
- 15 20 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass ein Anwender zum Formulieren von Bedingungen beliebig zwischen den Darstellungsformen textuelle Sprache, Kontaktplan "KOP" und/oder Funktionsplan "FUP" wechseln kann. Dadurch, dass auf der Structured Text-Ebene IEC 6-1131 als textuelle Sprache verwendet wird, können auch andere Darstellungsformen von IEC 6-1131 neben der textuellen Sprache, nämlich Kontaktpläne und/oder Funktionspläne, verwendet werden. Ein Anwender hat somit die Flexibilität, innerhalb dieser Sprachen der SPS-Welt, nämlich Structured Text, Kontaktplan oder Funktionsplan, beliebig zu wechseln. Diese Flexibilität ist insbesondere für die Formulierung von Bedingungen ein großer Vorteil für den Anwender, denn er kann sich die Darstellungs- bzw. Beschreibungsform wählen, in der er die meiste Erfahrung hat, oder die dem zugrunde liegenden Problem angemessen ist. Üblicherweise verwendet ein Anwender für die Darstellung von binären Verknüpfungen Kontaktpläne und/oder

Funktionspläne und für die Formulierung von arithmetischen Berechnungen Structured Text.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt  
5 darin, dass in der Flow Chart-Notation als Sprachelemente mindestens eine Schleife und/oder mindestens eine Parallelverzweigung vorhanden sind. In den heutzutage gängigen Flow Chart-Editoren werden Schleifen und oft auch Verzweigungen mit Hilfe von Sprungmarken dargestellt. Durch die Verwendung  
10 von Sprüngen (Goto-Problematik!) und die dazugehörigen Zielmarken wird die Programmgestaltung aber sehr unübersichtlich und schwer nachvollziehbar. Dadurch, dass dem Anwender als eigene Sprachelemente Schleifen und Parallelverzweigung zur  
15 Verfügung stehen, wird die Programmerstellung und auch die Lesbarkeit der Programme erheblich vereinfacht.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt  
darin, dass innerhalb der jeweiligen Parallelverzweigung die  
einzelnen Befehle im selben Interpolatortakt gestartet werden.  
20 Dadurch, dass alle Zweige des Sprachkonstrukts Parallelverzweigung im selben Interpolatortakt bedient werden, ist eine quasi parallele Abarbeitung der in den einzelnen Zweigen des Parallelverzweigungs-Konstrukts enthaltenen Befehle möglich. Neben der sequentiellen wird somit auch die parallele  
25 Abarbeitung von Befehlen ermöglicht und durch adäquate Sprachmechanismen in der Programmierumgebung für den Anwender unterstützt.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt  
30 darin, dass in der Flow Chart-Notation die Funktionsblöcke durch Maskeneingabe parametriert werden. Dadurch wird für den Anwender die Eingabe von Parametern in einer übersichtlichen und leicht verständlichen Form ermöglicht. Für jeden Typ von Funktionsblock existieren Standardmasken, die einem Anwender  
35 nur die für den aktuellen Typ möglichen Parametereingaben erlauben. Die Gefahr von fehlerhaften Eingaben wird durch diese Kontextsensitivität reduziert.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass in der Flow Chart-Notation Funktionsblöcke zu Modulen zusammengefasst werden, die wiederum als Funktionsblöcke erscheinen. Dadurch wird die Übersichtlichkeit des Programmablaufs im Flow Chart für den Anwender erhöht. Ein Anwender kann nämlich logisch zusammengehörende Funktionsblöcke zu einem Modul zusammenfassen und kapseln, wobei dieses Modul wiederum als Funktionsblock im Flow Chart-Editor, d.h. als Icon, erscheint. Durch diesen Mechanismus der Zusammenfassung und Kapselung wird aber nicht nur die Übersichtlichkeit im Ablauf erhöht, auch der Programmablauf lässt sich dadurch strukturieren.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass in der Flow Chart-Notation Module ineinander geschachtelt werden. Das heißt, ein Modul kann wiederum als Element ein oder mehrere Module enthalten. Module können sozusagen wiederum als Unterprogramme in anderen Modulen verwendet werden, dadurch wird die Übersichtlichkeit und die Struktur des Programmablaufs im Flow Chart erhöht.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass in der Flow Chart-Notation dem Anwender in den Funktionsblöcken für die Variablenzuweisung jeweils mehrere Zuweisungen möglich sind. Dadurch, dass der Anwender in einem Funktionsblock, d.h. in einem Icon, mehrere Variablenzuweisungen nacheinander eingeben kann und nicht für jede Variablenzuweisung einen neuen Funktionsblock benötigt, wird zum einen die Übersichtlichkeit erhöht, zum anderen wird aber auch das Programmierprinzip der hohen Kohäsion unterstützt, da der Anwender seine Variablenzuweisungen, die sinnvollerweise zu diesem Funktionsblock gehören, auch in diesem einen Funktionsblock gebündelt vornehmen kann.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass in der Flow Chart-Notation die Funktionsblöcke, die Funktionen repräsentieren, die eine Zeitdauer beanspru-

chen, Weiterschaltbedingungen enthalten. Funktionen, die eine Zeitspanne beanspruchen, sind z.B. Referenzpunktfahren, Beschleunigen oder Achspositionieren. Solche Funktionen bzw.

ihr Zusammenwirken können Anwender mit Hilfe der Weiterschaltbedingungen synchronisieren. Einem Anwender steht somit

mit Hilfe der Weiterschaltbedingungen ein Synchronisationsmechanismus zur Verfügung, der es ihm erlaubt, komplexe Bewegungen und Zusammenhänge mehrerer Achsen zu synchronisieren.

- 10 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die grafischen Elemente des Flow Charts automatisch positioniert werden. Wenn ein Anwender ein neues Icon im Flow Chart-Editor darstellen will, wird es automatisch an der Stelle positioniert, die als nächstes dem logischen Programmablauf entspricht. Dadurch, dass ein Anwender die generierten Icons nicht selbst positionieren muss, wird seine Arbeitseffizienz gesteigert.

20 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die Icons des Flow Charts automatisch miteinander verbunden werden. Auch hierin liegt eine Steigerung der Arbeitseffizienz des Anwenders, da er die Icons nicht nachträglich per Hand miteinander verbinden muss.

- 25 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass das Flow Chart in der Anzeige verkleinert oder vergrößert dargestellt wird. Durch diese Zoom-Funktionalität wird für den Anwender die Übersichtlichkeit der Diagramme erhöht und außerdem kann er bestimmte Programmabläufe, die ihn momentan interessieren, durch Vergrößerung graphisch hervorheben.

- 35 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass durch Markierungen in der textuellen Sprache eine Rückübersetzung in Flow Chart-Notation möglich ist. Durch die Verwendung von syntaktischen und geometrischen Informationen, die in Form von Markierungen erfolgen, ist es möglich, aus

der textuellen Sprache in die Flow Chart-Notation eine Rückübersetzung vorzunehmen. Diese Rückübersetzungsmöglichkeit hat für den Anwender den Vorteil, dass Änderungen, die auf der Ebene der textuellen Sprache eingegeben werden, unmittelbar im Flow Chart-Editor in der Flow Chart-Notation nachgezogen werden können und somit für den Anwender in den Flow Chart-Diagrammen sichtbar sind. Solche rückübersetzten Programme kann der Anwender dann auf der Graphikebene mit Hilfe des Flow Chart-Editors weiterbearbeiten und daraus im weiteren Vorgehen Steuerungscode erzeugen.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die Schritte a) bis c) in einem Sammelschritt ausgelöst werden. Damit muss der eigentliche Debug-Vorgang nach der Vorbereitung und der Zuordnung der Suspend-Befehle zu den grafischen Elementen nicht explizit vom Anwender in einem separaten Arbeitsschritt gestartet werden.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass während der Abarbeitung des Flow Chart Programms das jeweils aktuelle grafische Element optisch auf der Anzeigeeinrichtung gekennzeichnet wird. Ein Anwender kann somit im Flow Chart den Programmablauf verfolgen und dabei ein positives, aber auch ein negatives Verhalten des Programmes visualisiert erkennen.

Die wesentlichen mit der Erfindung erzielten Vorteile bestehen also insbesondere darin, dass ein Anwender bei der Fehlersuche und beim Programmtest auf der Abstraktionsebene von Flow Charts in drei Aspekten adäquat unterstützt wird: Programmbeobachtung, Einzelschrittabarbeitung (Single Step) und Einsatz von parametrierbaren Breakpoints.

Weitere Vorteile der Erfindung bestehen insbesondere darin, dass aus Unterprogrammen, die in der textuellen Sprache vorliegen, für den Flow Chart-Editor Icons generiert werden, die die Funktionsschnittstelle der entsprechenden Unterprogramme

automatisch enthalten. Wenn ein OEM-Kunde bereits Unterprogramme in der textuellen Sprache erstellt hat, so können diese Unterprogramme automatisch durch entsprechende Icons den Sprachvorrat des Flow Chart-Editors erweitern.

5

Ein weiterer sehr großer Vorteil liegt darin, dass ein Anwender in einer einheitlichen Programmierumgebung sowohl Bewegungssteuerungsaufgaben (Motion Control) und Prozesssteuerungsaufgaben (SPS-Aufgaben) in einer jeweils angemessenen Form programmieren kann. Weiterhin ist von Vorteil, dass die Programmierumgebung sich projektsensitiv verhält, d.h. dass dem Anwender in Abhängigkeit von der zugrunde liegenden Hardware bzw. Maschinenprojektierung zusätzliche dedizierte Sprachelemente zur Verfügung gestellt werden.

15

Ein weiterer Vorteil liegt darin, dass der Anwender sowohl für die sequentielle als auch für die zyklische Programmierung der Steuerungsabläufe unterstützt wird. Dadurch, dass eine geschachtelte Modulbildung von Funktionsblöcken zur Verfügung steht, hat der Anwender den Vorteil, die Übersichtlichkeit und die Struktur seiner Programme zu erhöhen, da er die Designkriterien, Lokalität und hohe Kohäsion sehr leicht umsetzen kann.

20

Ein Ausführungsbeispiel der Erfindung ist in der Zeichnung dargestellt und wird im folgenden erläutert.

Dabei zeigen:

30 FIG 1

in einer Schemadarstellung ein Engineering-System, das zugehörige Run-Time-System und den zu steuern- den technischen Prozess,

FIG 2

zeigt in einem Übersichtsbild Elemente des Enginee- ring-Systems und der Steuerung sowie ihre Beziehun- gen untereinander,

35

FIG 3 zeigt ebenfalls in Form eines Übersichtsbildes den programmtechnischen Zusammenhang zwischen Elementen des Engineering-Systems und des Run-Time-Systems,

5 FIG 4 zeigt ein einfaches Diagramm in Flow Chart-Notation,

FIG 5 zeigt ein komplexes Diagramm in Flow Chart-Notation mit den Kontrollstrukturen while und if,

10

FIG 6 zeigt ebenfalls ein komplexes Diagramm in Flow Chart-Notation mit dem Sprachkonstrukt Parallelverzweigung (sync),

15 FIG 7 zeigt eine Parametriermaske für den Befehl "positiere Achse",

FIG 8 zeigt in einem Übersichtsbild, wie der Sprachvorrat des Flow Chart Editors erweitert wird,

20

FIG 9 zeigt eine Auswahl von Sprachelementen (so genannten Icons) des Flow Chart-Editors.

In der Darstellung gemäß FIG 1 wird in Form eines Strukturbildes gezeigt, dass die Steuerung eines technischen Prozesses TP über das Run-Time-System RTS einer industriellen Steuerung erfolgt. Die Verbindung zwischen dem Run-Time-System RTS der Steuerung und dem technischen Prozess TP geschieht

25 bidirektional über den Ein-/Ausgang EA. Die Programmierung der Steuerung und damit das Festlegen des Verhaltens des Run-

Time-Systems RTS geschieht im Engineering-System ES. Das Engineering-System ES enthält Werkzeuge für die Konfigurierung, Projektierung und Programmierung für Maschinen bzw. für die Steuerung technischer Prozesse. Die im Engineering-System er-

30 stellten Programme werden über den Informationspfad I1 in das Run-Time-System RTS der Steuerung übertragen. Bezüglich seiner Hardware-Ausstattung besteht ein Engineeringssystem ES üb-

licherweise aus einem Computersystem mit Grafikbildschirm (z.B. Display), Eingabehilfsmitteln (z.B. Tastatur und Maus), Prozessor, Arbeits- und Sekundärspeicher, einer Einrichtung für die Aufnahme computerlesbarer Medien (z.B. Disketten, CDs) sowie Anschlusseinheiten für einen Datenaustausch mit anderen Systemen (z.B. weiteren Computersystemen, Steuerungen für technische Prozesse) oder Medien (z.B. Internet). Eine Steuerung besteht üblicherweise aus Eingabe- und Ausgebeeinheiten, sowie aus Prozessor und Programmspeicher.

10

Das Run-Time-System RTS enthält einen Tasksteuermechanismus, der beim Debugging der Flow Charts verwendet wird. Der Tasksteuermechanismus kann z.B. vom Engineering System ES mit Informationen versorgt werden. Insbesondere ist es Aufgabe des Tasksteuermechanismus die ResUME-Befehle zu bedienen, um eine Fortsetzung des Programmablaufs zu bewirken.

20

In der Darstellung gemäß FIG 2 werden in Form eines Übersichtsbildes Elemente des Engineering-Systems und der Steuerung sowie ihr Zusammenspiel dargestellt. Dabei werden die einzelnen Elemente in Form von Rechtecken dargestellt, die im Engineering-System enthaltene Datenablage wird in Form des üblicherweise verwendeten Datenspeichersymbols dargestellt. Durch Pfeile (unidirektional oder bidirektional) wird der datenlogische bzw. der ablauflogische Zusammenhang zwischen den Elementen dargestellt. Die obere Hälfte von FIG 2 zeigt die Elemente des Engineering-Systems, nämlich den MCC-Editor, den ST-Compiler mit Programmierergebnung, den Konfigurations-Server KS und die Maschinenprojektierung sowie eine Datenablage. Die Zugehörigkeit dieser Elemente zum Engineering-System wird durch Umrandung dargestellt. Die Steuerung beinhaltet den Codeumsetzer und die Programmverarbeitung. Auch die Elemente der Steuerung, die sich im unteren Abschnitt von FIG 2 befinden, sind umrandet. Sowohl das Engineering-System als auch die Steuerung können noch weitere Elemente beinhalten, sie sind aber aus Gründen der Übersichtlichkeit nicht dargestellt.

Im MCC-Editor (MCC steht für Motion Control Chart) werden die graphischen Programmabläufe erzeugt. Die Sprachelemente des Editors, d.h. die Icons, können z.B. über eine Befehlsleiste im Bildschirm, die mit Hilfe einer Maus bedient wird oder  
5 denkbare andere Eingabehilfsmittel mit dem Editor erzeugt und dargestellt werden. Ein Anwender kann mit Hilfe des MCC-Editors Funktionsblöcke (Icons) und Kontrollstrukturen zu einem Flow Chart verknüpfen, d.h. er kann den MCC-Editor als graphisches Programmier-Tool für die Erstellung von Programmen für Bewegungssteuerungen und/oder Prozesssteuerungen verwenden. Aus dem Flow Chart wird ein textuelles Programm bzw.  
10 eine textuelle Sprache (üblicherweise Structured Text nach IEC 6-1131) erzeugt. Dieser Structured Text-Code (ST-Code) wird vom Structured Text-Compiler (ST-Compiler, der Teil der Programmierumgebung ist) in einen prozessorunabhängigen Zwischencode kompiliert. Dieser Zwischencode wird auf die Steuerung geladen und dort vom Codeumsetzer in ablauffähigen Prozessorcode umgesetzt. Dieser wird von der Programmverarbeitung innerhalb der Steuerung zum Ablauf gebracht. Durch die  
15 unidirektionalen Pfeile im linken Abschnitt von FIG 2 werden die Schritte der Code- bzw. Programmumsetzung dargestellt. Parallel zu den drei von oben nach unten verlaufenden unidirektionalen Pfeilen, die diese Umsetzung darstellen, verlaufen jeweils zwischen den Elementen MCC-Editor, ST-Compiler,  
20 Codeumsetzer und Programmverarbeitung drei bidirektionale Pfeile, die Debug-Schnittstellen bzw. die Möglichkeit einer Programmbeobachtung darstellen. Zwischen Programmverarbeitung und Codeumsetzer existiert eine Debug-Schnittstelle auf Prozessorcode-, d.h. auf Objektcodeebene, eine weitere Debug-  
25 Schnittstelle existiert zwischen dem Codeumsetzer und dem ST-Compiler, diese Debug-Schnittstelle befindet sich auf Zwischencodeebene. Zwischen dem ST-Compiler und dem MCC-Editor befindet sich eine weitere Debug- bzw. Programmbeobachtungsschnittstelle auf Ebene von Structured Text (ST-Code).  
30

35

Dadurch ist für einen Anwender das Debuggen von Programmen in der jeweiligen Codeebene in der dazugehörigen Abstraktions-

stufe möglich. Ein Anwender ist somit in der Lage je nach Ausbildungsstand oder Erfahrung sich eine geeignete Abstraktionsebene zum Debuggen zu wählen. Außerdem ist es bekannt, dass auf jeder Codeebene bestimmte Typen von Fehlern mehr oder weniger häufig auftauchen. Durch die Möglichkeit des Debuggens auf unterschiedlichen Codeebenen kann ein Anwender gezielt nach Fehlern suchen, die für die jeweilige Codeebene typisch sind. Die Fehlersuche bzw. Fehlerlokalisierung wird dadurch effizienter.

10

In FIG 2 sind als weitere Elemente des Engineering-Systems die Maschinenprojektierung und ein Konfigurations-Server KS dargestellt. In der Maschinenprojektierung wird mit Hilfe geeigneter Werkzeuge die Auslegung der Hardware bzw. der zugrunde gelegten Maschine vollzogen. Das heißt, in der Maschinenprojektierung wird z.B. festgelegt, welche Achstypen in welcher Anzahl physikalisch vorhanden sind. Diese Maschineninformationen werden über den Konfigurations-Server KS in den MCC-Editor eingespeist. Die Übertragung dieser Informationen wird durch die unidirektionalen Pfeile I2 und I3 dargestellt. Weiterhin beinhaltet der Konfigurations-Server KS weitere relevante Konfigurationsinformationen für das System, die z.B. auch für die Lizenzierung von zugehörigen Softwarekomponenten verwendet werden können.

25

Eine Datenablage DA, dargestellt durch das gängige Datenspeichersymbol, beinhaltet drei Aspekte: Zum einen das vom MCC-Editor für ein Flow Chart erzeugtes Objektmodell, als zweites den dazugehörigen Structured Text und der dritte Inhalt der Datenablage DA ist der aus dem Structured Text generierte Zwischencode. Die Datenablage DA steht in bidirektonaler Verbindung zum MCC-Editor und ST-Compiler, dargestellt durch die bidirektonalen Informationspfeile I4 und I5.

35 Die Darstellung gemäß FIG 3 zeigt als Übersichtsbild die vorhandenen Abstraktionsebenen aus Sicht des Programmcodes. Die unterschiedlichen Programmcode-Ebenen sind als Rechtecke dar-

gestellt. Die oberste Ebene ist die MCC-Ebene, in der die Flow Chart-Programme erzeugt werden. Die nächstuntergeordnete Codeebene ist die Structured Text-Ebene ST. In die ST-Ebene gelangt man aus der MCC-Ebene durch eine entsprechende Code-5 generierung, dargestellt durch einen Pfeil vom MCC-Block zum ST-Block. Unterhalb der Structured Text-Ebene ST liegt die Zwischencode-Ebene. Durch einen Compiler wird aus dem Structured Text-Programm ein prozessorunabhängiger Zwischencode kompiliert, dargestellt durch den Pfeil vom ST-Block zum Block mit dem Namen Zwischencode. Unterhalb der Zwischencode-10 Ebene liegt die unterste Codeebene, nämlich die Objektcode-Ebene, die den ablauffähigen Prozessorcode beinhaltet. Aus dem Zwischencode wird über einen Umsetzer der Objektcode erzeugt, ebenfalls dargestellt durch einen Pfeil vom Zwischen-15 codeblock zum Objektcodeblock. Von der Objektcode-Ebene gehen rechtwinklig abgewinkelte Pfeile zurück zur Structured Text-Codeebene ST und zur Flow Chart-Ebene MCC. Dadurch ist ange-deutet, dass auf diesen Ebenen Test- und Programmverfolgungs-aktivitäten stattfinden können, auf der Basis des Objekt-20 codes. Durch den fetten Doppelpfeil zwischen der MCC- und der ST-Ebene wird angedeutet, dass zwischen diesen beiden Ebenen Aufrufe, Task-Steuerbefehle und Variablenaustauschfunktionen möglich sind. Die gestrichelte Linie in FIG 3 zeigt die Grenze zwischen dem Engineering System ES und dem Run-Time-System 25 RTS der Steuerung (S; FIG 2) an. Die Grenze verläuft durch die Zwischencode-Ebene, alles, was oberhalb der gestrichelten Linie ist, gehört zum Engineering System ES, alles, was un-terhalb der gestrichelten Linie stattfindet, gehört zum Run-Time-System RTS.

30 Weiterhin wird in FIG 3 gezeigt, wie ein Programmierer oder Anwender (am linken Bildrand dargestellt durch ein stilisiertes Strichmännchen) im Engineering System ES Eingaben ein-bringen kann. Er kann zum einen auf der MCC-Ebene mit Hilfe 35 der graphischen Programmierung Flow Charts erzeugen, zum an-deren kann er auf der Structured Text-Ebene ST durch eine textuelle Programmierung Programme erstellen. Beide Eingabe-

möglichkeiten sind durch Pfeile vom Strichmännchen zum MCC-Block bzw. zum ST-Block dargestellt.

Darstellung gemäß FIG 4 zeigt einen einfachen Programmablauf

5 für die Programmierung von Achsbewegungen. Jedes Flow Chart beginnt mit einem Startknoten und endet mit einem Endeknoten.

Diese Programmbegrenzungssymbole tragen die Bezeichnung "Start" bzw. "Ende". Start- und Endesymbole werden jeweils durch ein Rechteck dargestellt, dessen Stirnseiten durch zwei Halbkreise ausgebildet sind. Die Programmbefehle werden durch Rechtecke dargestellt, die einen Bezeichner und ein graphisches Symbol beinhalten, welches den hinterlagerten Befehl repräsentiert.

15 Die Flow Chart-Symbole werden üblicherweise über eine Eingabeleiste mit Hilfe einer Maus im Flow Chart-Editor erzeugt, wobei auch andere Eingabehilfsmittel wie z.B. ein Touch Pad denkbar sind. Alternativ wäre auch eine Bedienung über Tastatur mit oder ohne Maus möglich.

20

Die Flow Chart-Symbole werden vom Flow Chart-Editor defaultmäßig untereinander ausgerichtet und durch eine Linie miteinander verbunden.

25 Im Flow Chart nach FIG 4 wird nach dem Start eine Gleichlaufachse freigeschaltet, danach wird auf ein Synchronisierungssignal gewartet und als nächster und letzter Befehl des Flow Charts wird für eine Gleichlaufachse eine Kurvenscheibe eingeschaltet. Die Befehlsequenz von FIG 4 wird beendet durch

30 das Endesymbol.

Wenn ein Anwender das vorliegende Flow Chart Debuggen (Fehlersuche, Fehlerlokalisierung, Kontrolle des Programmverhaltens etc.) will, wird er einen Debug-Modus starten, der bewirkt, dass den einzelnen grafischen Elementen des Flow Charts Suspend-Befehle zugeordnet werden. Er kann dann im Single-Step-Modus oder im Breakpoint-Modus das Debugging

durchführen. Im Single-Step-Modus erfolgt der Programmablauf nach dem Starten des Debuggers (ein Debugger ist als Dienstprogramm der Programmierumgebung, d.h. somit auch des Engineeringsystems (ES; FIG1, FIG2, FIG3) anzusehen) automatisch von Suspend-Befehl zum nächsten Suspend-Befehl. Bei jedem erreichten Suspend-Befehl wird der Programmablauf gestoppt. Das Erreichen eines Suspend-Befehls und des zugehörigen grafischen Elements wird dem Anwender visuell (z.B. durch speziellen Cursor oder farbiger Kennzeichnung) angezeigt. Das "Lösen" der Suspend-Befehle geschieht durch entsprechende Resume-Befehle. Der Anwender kann dadurch die Fortsetzung des Programmablaufs bewirken. Ein Fortsetzen des Programmablaufs kann auch durch einen Tasksteuermechanismus des Run-Time-Systems erfolgen.

Einem Suspend-Befehl können aber auch Bedingungen, Variablenwerte oder Speicheradressen zugeordnet werden. Der Programmablauf wird dann durch den Wert dieser Zuordnungen bestimmt. Dadurch kann mit Hilfe der Suspend-Befehle ein Debugging auch im Breakpoint-Modus erfolgen.

Die Darstellung gemäß FIG 5 zeigt ein komplexes Flow Chart mit Kontrollstrukturen für eine While-Schleife und für das If-Konstrukt. Das While- und das If-Konstrukt werden jeweils durch sechseckige, wabenförmige Symbole dargestellt. Ansonsten werden im Programmablauf, wie er in FIG 5 dargestellt ist, die gleichen Typen von Symbolen verwendet, wie sie schon aus FIG 4 bekannt sind. Auch das Flow Chart nach FIG 5 beginnt mit dem Start- und endet mit dem Endesymbol. Unmittelbar nach dem Startknoten folgt ein Befehl, der die Task "motion\_3" startet. Dieser Befehl ist vom Typ "Starte Task". Das Rechteck für diesen Befehl enthält deshalb auch das zugehörige entsprechende Symbol, welches das Starten einer Task darstellt. Als nächstes im Programmablauf, wie er in FIG 5 dargestellt ist, folgt das sechseckige wabenförmige While-Konstrukt. Solange die im While-Konstrukt angegebene Bedingung true ist, werden die auf das While-Konstrukt folgenden

Befehle zyklisch nacheinander ausgeführt. Das Ende der Befehlsfolge einer While-Schleife wird dargestellt durch einen abgewinkelten Pfeil, der vom letzten Symbol des While-Konstrukts (in FIG 5 ist dies der Befehl vom Typ "Getriebegleichlauf aus", bezogen auf eine Gleichlaufachse) von unten abgeht und auf der linken Seite von FIG 5 zurück in das While-Konstrukt mündet. Ist die Bedingung im While-Konstrukt nicht mehr erfüllt, dann wird die Befehlsfolge, die zum While-Konstrukt gehört, nicht mehr ausgeführt. In FIG 5 wird dies dargestellt durch eine rechtwinklige Verbindungsleitung, die das While-Symbol auf der rechten Seite verlässt und rechter Hand die zum While-Symbol gehörende Symbolbefehlsfolge umgeht und in auf das dieser Befehlsfolge unmittelbar folgende Symbol einmündet, in FIG 5 ist dies das Ende-Symbol.

Wenn aber die While-Bedingung erfüllt ist, wird folgende Befehlsfolge abgearbeitet: Unmittelbar nach dem While-Konstrukt folgt ein Befehl, der das Warten auf eine Bedingung repräsentiert. Auch dieser Befehl enthält ein entsprechendes mnemotechnisches grafisches Symbol, das den Wartevorgang graphisch darstellt. Als nächstes folgt ein Befehl, der die Task "motion\_2" startet. Auch dieser Befehl ist vom Typ "Starte Task" und enthält das entsprechende graphische Symbol. Nach diesem Befehl folgt das If-Konstrukt, das genauso wie das While-Konstrukt durch ein sechseckiges, wabenförmiges Symbol dargestellt wird. Ist die If-Bedingung (in FIG 5 dargestellt durch "error <> 0"), erfüllt dann wird im True-Zweig die Befehlsfolge weiter abgearbeitet, ansonsten, wenn die Bedingung nicht erfüllt ist, wird die Befehlsfolge im False-Zweig weiter abgearbeitet. Im True-Zweig der If-Bedingung folgt als nächster Befehl ein Befehl, der die Task "motion\_2" stoppt. Dieser Befehl ist vom Typ "Stoppe Task". Darauf folgt ein Befehl, der die Task "motion\_3" stoppt. Auch dieser Befehl ist vom Typ "Stoppe Task". Diese Befehle werden außerdem durch dazugehörige entsprechende Symbole repräsentiert. Als nächstes in der Befehlsfolge kommen zwei "Stoppe Achs"-Befehle. Im ersten solchen Befehl wird eine Drehzahlachse gestoppt, im

darauffolgenden eine Positionierachse, auch diese "stoppe Achs"-Befehle werden durch dazugehörende entsprechende grafische Symbole repräsentiert. Der nächste und zugleich der letzte Befehl in FIG 5 bezieht sich auf eine Achse mit dem  
5 Namen "Gleichlaufachse", nämlich auf das Ausschalten des Getriebegleichlaufes ("Getriebegleichlauf aus"), auch dieser Befehl wird durch ein entsprechendes graphisches Symbol repräsentiert. Die Symbole des Flow Charts sind durch Linien miteinander verbunden, womit der Programmablauf dargestellt  
10 wird. Von diesem Befehl, der den letzten Befehl im While-Konstrukt darstellt, geht ein rechtwinklig abgewinkelter Pfeil zurück zu diesem While-Konstrukt. Dadurch wird das zyklische Abarbeiten der Befehlsfolge dargestellt. Im While-Konstrukt wird geprüft, ob die Bedingung erfüllt ist. Ist sie  
15 erfüllt oder weiterhin erfüllt, wird die Befehlsfolge noch einmal durchlaufen. Ist sie nicht erfüllt, wird das While-Konstrukt verlassen und, wie beispielhaft in FIG 5 dargestellt, mit dem Ende-Symbol fortgefahren, d.h. der durch das Flow Chart dargestellte Programmablauf wird beendet.

20

Auch dem Flow Chart gemäß FIG 5 können zum Zweck des Debuggens Suspend-Befehle zugeordnet werden.

25

Die Darstellung gemäß FIG 6 zeigt ebenfalls ein komplexes Diagramm in Flow Chart-Notation mit dem Sprachkonstrukt Parallelverzweigung (sync). In FIG 6 folgt auf das Start-Symbol ein Befehl, der sich auf eine Drehzahlachse bezieht, nämlich "Achsfreigabe schalten". Auch für diesen Befehl wird im Befehlsrechteck ein graphisches Symbol angegeben, das diesen Befehl repräsentiert. Danach folgt wiederum ein Befehl vom Typ "Achsfreigabe schalten", diesmal aber bezogen auf eine Positionierachse, auch hier ist das dazugehörige entsprechende Symbol angegeben. Der nächstfolgende Befehl ist ein Synchronisationsbefehl "warte auf Signal", in FIG 6 mit "Auto"  
30 bezeichnet und mit dem entsprechenden Symbol versehen.  
35

Als nächstes Symbol in FIG 6 folgt das Symbol für die Parallelverzweigung (sync). Dieses Symbol wird ebenfalls wie das While- oder das If-Konstrukt durch ein sechseckiges, wabenförmiges graphisches Element dargestellt. Alle Befehle, die 5 in dem Sektor unmittelbar unter dem Symbol für die Parallelverzweigung angeordnet sind, werden im selben Interpolatortakt gestartet. In FIG 6 sind dies die Befehle "Positioniere Achse", bezogen auf eine Positionierachse (dieser Befehlstyp beinhaltet auch das zugehörige entsprechende graphische Symbol) und ein Befehl vom Typ "Setze Ausgang". Der Befehlstyp "Setze Ausgang" ist ebenfalls durch ein Rechteck dargestellt, dieses Rechteck enthält die Adresse des Ausgangs (%QB40) und 10 das entsprechende Symbol für diesen Setz-Befehl (S steht für Set). Die Befehle, die zu einem Parallelverzweigungssymbol gehören, d.h. die innerhalb desselben Interpolatortakts gestartet werden, sind mit dem Parallelverzweigungssymbol nach oben mit einer Linie verbunden und nach unten sind sie mit einer Doppellinie verbunden. Diese waagrechte Doppellinie zeigt an, dass die parallele Abarbeitung wieder aufgehoben 15 ist und dass mit der Bearbeitung des nachfolgenden Befehls so lange gewartet wird, bis alle Aktionen in der Parallelverzweigung beendet sind. Sie ist somit auch das Ende-Symbol des Parallelverzweigungskonstrukts. Als nächstes folgt ein Befehl 20 vom Typ "Drehzahlvorgabe", der sich auf eine Drehzahlachse bezieht. Darauf folgen zwei Befehle vom Typ "Positioniere Achse", die sich jeweils auf Positionierachsen beziehen. Darauf 25 folgt wieder ein Befehl vom Typ "Stoppe Achse", der sich auf eine Drehzahlachse bezieht. Die Rechtecke, die diese genannten Befehle darstellen, beinhalten natürlich auch wieder entsprechende dazugehörige graphische Symbole. Nach dem Befehl 30 vom Typ "Stoppe Achse", der sich auf die schon genannte Drehzahlachse bezieht, folgt das Ende-Symbol.

Die hier dargestellte Art der Flow Chart-Programmierung unterstützt unterschiedliche Arten der Programmierung. Zum einen wird durch das Parallelverzweigungssymbol mit dem Starten der dazugehörigen Befehle in einem Interpolatortakt eine mehr 35

oder weniger echte Parallelität erreicht, d.h. die Programmierung paralleler Threads wird unterstützt, und ihre dazugehörige Abarbeitung wird ermöglicht. Zum anderen wird die zyklische Programmierung, d.h. auch die zyklische Programmabarbeitung, unterstützt, es kann nämlich dargestellt werden,  
5 dass aufeinander folgende Befehle nur angestoßen werden, wobei aber jeweils nicht auf die Abarbeitung des vorhergehenden Befehls gewartet werden muss. Aber auch die Programmierung und die Darstellung solcher sequentiellen Abläufe wäre möglich,  
10 dass nämlich bei Anstoß eines Befehls auf die Abarbeitung dieses Befehls gewartet wird, bis der nächste Befehl angestoßen und abgearbeitet wird. Die hier vorgestellte Flow Chart-Programmierung ist somit für einen Anwender sehr flexibel anwendbar und für unterschiedliche Applikationen einsetzbar.  
15

Auch dem Flow Chart gemäß FIG 6 können zum Zweck des Debuggens Suspend-Befehle zugeordnet werden.

20 Die Darstellung gemäß FIG 7 zeigt eine Parametriermaske für den Flow Chart-Befehl "Positioniere Achse". Oben links im oberen Balken der Parametriermaske steht die Bezeichnung des entsprechenden Befehls, in diesem Fall "Positioniere Achse". Der obere Balken beinhaltet auf seiner rechten Seite zwei  
25 Schalter, ein Schalter mit einem Fragezeichen versehen beinhaltet eine Online-Hilfe, der zweite Schalter, mit x versehen, wird für das Schließen der Maske verwendet. Die Parametriermaske beinhaltet unterschiedliche Eingabesektoren. Im obersten Eingabesektor kann die entsprechende Achse ausgewählt werden mit Hilfe eines Eingabemenüs (dargestellt durch einen Eingabeknopf mit einem kleinen auf dem Kopf stehenden Dreieck) können im Eingabefenster die entsprechenden Achsen ausgewählt werden. In diesem obersten Sektor ist links oben auch das zu diesem Befehl dazugehörige graphische Symbol angegeben, nämlich ein auf dem Kopf stehendes Dreieck mit der Spitze nach unten, das waagrecht mittig mit einer dunklen Linie versehen ist, wobei an den Enden dieser Linie jeweils  
30  
35

nach unten abgeschrägte weitere kleine Linien angebracht sind. Der nächste und größte Sektor der Parametriermaske stellt die Möglichkeiten dar, Parameter einzugeben. Die Parameter sind, je nach Befehl, unterschiedlich. Sie werden über benannte Reiter, die auf einer Reiterleiste angeordnet sind, wie in gängigen Programmoberflächen üblich, logisch sortiert. Die erste Seite (in FIG 7 ist diese Seite durch den Reiter "Parameter" aufblendbar) trägt üblicherweise die Parameter, die unbedingt zur Parametrierung des Befehls angegeben werden müssen. Für den Befehl "Positioniere Achse" ist ein unbedingter Parameter z.B. die Zielposition einer Achsbewegung.

Die Anzahl und Bedeutung der Reiter ist befehlsabhängig unterschiedlich. In FIG 7 ist dargestellt, dass für den Befehl "Positioniere Achse" neben dem Reiter "Parameter" noch ein Reiter "Dynamic" vorhanden ist. Mit diesem Reiter können für die Beschreibung des dynamischen Verhaltens Eingaben zu Ruck und Beschleunigung sowie zum Geschwindigkeitsprofil gemacht werden. Diese Eingaben können über Eingabefelder und dazugehörigen Menüs gemacht werden. In Abbildung von FIG 7 wurde als Geschwindigkeitsprofil die Trapezform gewählt. Diese Form wurde auch graphisch in der Mitte dieses Eingabesektors stilisiert dargestellt. Im unten darauffolgenden Eingabesektor der Parametriermaske können weitere Eingaben, z.B. für das Übergangsverhalten, gemacht werden. Im Beispiel von FIG 7 wurde für das Übergangsverhalten "Ablösen" eingegeben. Zusätzlich können Wartebedingungen eingegeben werden, indem das Kästchen "Warten" mit einem Häkchen versehen wird. Zu diesem Synchronisieren können in einem dazugehörenden Eingabefenster weitere Eingaben gemacht werden. Im Beispiel nach FIG 7 wurde "Positionsfenster erreicht" dafür eingegeben. Auch diese Eingaben werden durch stilisiert dargestellte Achsprofile unterstützt. Das untere Ende einer Parametrierachse besteht aus vier Eingabeknöpfen, nämlich einem "OK"-Knopf, einem "Abbrechen"-Knopf, einem "Übernehmen"-Knopf und einem "Hilfe"-Knopf. Mit Hilfe dieser Eingabeknöpfe können Anwender entweder die Eingaben übernehmen, bestätigen, verwerfen oder die

Eingabehilfe aufrufen. Mit Hilfe der Wartebedingungen können durch einen Anwender sogenannte Weiterschaltbedingungen spezifiziert werden, die Funktionen (z.B. Referenzpunktfahren oder Achspositionieren) bzw. ihr Zusammenspiel synchronisieren.

Solche Parametriermasken existieren dediziert für alle Befehle, die mit Hilfe des Flow Chart-Editors eingegeben und bearbeitet werden können. Der Anwender wird also kontextsensitiv mit Hilfe dieser Parametriermasken bei der Programmierung seiner Bewegungs- und Steuerungsabläufe unterstützt.

Die Darstellung gemäß FIG 8 zeigt in einem Übersichtsbild, wie der Sprachvorrat FEV des Flow Chart Editors FE erweitert wird. In der Ausgangssituation stehen einem Anwender A1 (dargestellt durch ein Strichmännchen) die grafischen Elemente E1 bis Em des Flow Chart Editors FE zur Verfügung. Der Flow Chart Editor FE ist in der Darstellung gemäß FIG 8 links oben als Rechteck dargestellt, das angedeutet grafische Elemente beinhaltet. Der Flow Chart Editor FE ist Teil eines Engineering Systems ES1, dargestellt durch eine gestrichelte Linie. Das Engineeringsystem ES1 enthält noch weitere Elemente, die aber aus Gründen der Übersichtlichkeit nicht dargestellt sind. Der Sprachvorrat FEV des Flow Chart Editors FE, der in der Ausgangssituation die grafischen Elemente E1 bis Em enthält, ist rechts oben in FIG 8 als Rechteck dargestellt.

Die untere Hälfte von FIG 8 zeigt einen Anwender A2 (ebenfalls dargestellt durch ein Strichmännchen) der auf der Structured Text Ebene (ST; FIG 3) mit einem Engineering System ES2 arbeitet. Auf der Structured Text Ebene (ST; FIG 3) stehen dem Anwender A2 innerhalb des Structured Text Editors STED die Structured Text Sprachelemente STE1 bis STEn zur Verfügung, die den Sprachvorrat STEDV des Structured Text Editors darstellen. Auch der Sprachvorrat STEDV des Structured Text Editors STED ist als Rechteck dargestellt. Mit Hilfe der Sprachelemente STE1 bis STEn kann der Anwender A2 im Structu-

red Text Editor STED Unterprogramme STUP erstellen. Diese Unterprogramme STUP werden über einen Umsetzer (z.B. durch einen Compiler C) in grafische Sprachelemente des Flow Chart Editors FE umgesetzt.

5

In FIG 8 ist links unten skizzenhaft dargestellt, wie diese Generierung grafischer Elemente erfolgt. Beispielhaft erfolgt die Umsetzung innerhalb des Engineering Systems ES2. Das Structured Text Unterprogramm STUP (schematisch dargestellt durch eine Abfolge von Structured Text Elementen aus dem Sprachvorrat STEDV des Structured Text Editors wird durch den Compiler C (dargestellt durch ein Rechteck mit einer Diagonallinie) umgesetzt in das grafische Element En, das auch die Funktionsschnittstelle des ursprünglichen Structured Text Unterprogramms enthält. Der Umsetzvorgang (Structured Text Editor -> Compiler -> grafisches Element) ist dabei schematisch angedeutet durch zwei waagrecht verlaufende Pfeile. Durch den Zuordnungspfeil ZP ist angedeutet, dass das neu generierte grafische Element En den Sprachvorrat FEV des Flow Chart Editors FE erweitert und dem Anwender A1 für dessen Flow Chart Programmierung zur Verfügung steht.

Auch das Engineeringssystem ES2 enthält noch weitere Elemente, die aber aus Gründen der Übersichtlichkeit nicht dargestellt sind. Für den beschriebenen Mechanismus ist es außerdem vorstellbar, dass sich die Funktionalitäten der Engineering Systeme ES1 und ES 2 in einem einzigen Engineering System befinden können. Auch können die beiden Anwender A1 und A2 durch eine einzige Person repräsentiert werden.

30

Die Darstellung gemäß FIG 9 zeigt eine Auswahl von Sprachelementen (so genannten Icons) des Flow Chart-Editors. Diese Sprachelemente repräsentieren Befehle, die der Anwender bei der graphischen Programmierung im Flow Chart-Editor benutzen kann. Der MCC-Flow Chart-Editor unterstützt folgende Klassen von Befehlen und stellt für die einzelnen Befehle dieser Klassen jeweils entsprechende Symbole zur Verfügung:

- Start-Befehle, Stop-Befehle, Positionierbefehle, Gleichlauf- und Kurvenscheiben-Befehle, Messtaster und SW-Nocken-Befehle, Warte-Befehle, Tasksteuer-Befehle, Befehle für die Manipulierung von Variablen sowie weitere allgemeine Befehle. Außerdem
- 5 stellt der MCC-Flow Chart-Editor weitere grafischen Kontrollstrukturen für den grafischen Programmablauf zur Verfügung.

## Patentansprüche

1. Verfahren für das Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungssteuerungen, wobei ein Anwender mit einem Editor grafische Elemente, insbesondere Kontrollstrukturen und Funktionsblöcke, zu einem auf einer Anzeigeeinrichtung visualisierbaren Flow Chart (MCC) verknüpft, gekennzeichnet durch die folgenden Verfahrensschritte:
- 10 h) vom Anwender wird ausgehend vom Flow Chart ein Debug-Vorgang vorbereitet,
- 15 i) dadurch wird jedem grafischen Element (E1-En) ein Suspend-Befehl zugeordnet,
- j) der Debug-Vorgang wird gestartet,
- 20 k) der Programmablauf folgt bis zum unmittelbar folgenden Suspend-Befehl,
- l) dem Anwender (A1, A2) wird visualisiert, wo im Flow Chart die aktuelle Flow Chart befindlich ist,
- 25 m) der Anwender startet das Fortschalten zum nächst möglichen Suspend-Befehl,
- n) die Schritte d) bis f) werden solange fortgeführt, bis das Flow Chart-Ende erreicht ist.
- 30 2. Verfahren nach Anspruch 1,  
dadurch gekennzeichnet,  
dass ein durch einen Suspend-Befehl angehaltenes grafisches Element, bzw. die zu diesem Element gehörige Task mit einem  
35 Tasksteuermechanismus des Run-Time-Systems (RTS) fortgesetzt wird.

3. Verfahren nach Anspruch 1 oder 2,  
dadurch gekennzeichnet,  
dass der Anwender im Engineering System (ES, ES1, ES2) über  
den Tasksteuermechanismus des Run-Time-Systems (RTS) einen  
5 Resume-Befehl bedient, der den aktuellen Suspend-Befehl fort-  
schaltet.
4. Verfahren nach Anspruch 1, 2 oder 3,  
dadurch gekennzeichnet,  
10 dass der Tasksteuermechanismus des Run-Time-Systems (RTS)  
über vom Anwender im Engineering System (ES, ES1, ES2) vorbe-  
legbare Variablen in Form eines Breakpoint-Debuggings verwen-  
det wird.
- 15 5. Verfahren nach Anspruch 1, 2, 3 oder 4,  
dadurch gekennzeichnet,  
dass die Variablenvorbelegungen im Tasksteuermechanismus  
durch andere Programme des Run-Time-Systems (RTS) erfolgen.
- 20 6. Verfahren nach einem der vorstehenden Ansprüche,  
gekennzeichnet durch die aufeinander fol-  
genden Schritte:
- a) aus dem Flow Chart wird eine textuelle Sprache (ST) er-  
zeugt,  
25 b) die textuelle Sprache (ST) wird in einen prozessorunab-  
hängigen Zwischencode kompiliert,  
c) der prozessorunabhängige Zwischencode wird auf die Steu-  
erung geladen,  
30 d) der prozessorunabhängige Zwischencode wird in ablauffä-  
higen Prozessorcode umgesetzt.  
35 7. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,

dass dem Anwender auf Ebene der textuellen Sprache (ST) und/oder auf Ebene des Zwischencodes und/oder auf Ebene des Prozessorcodes jeweils eine Debug-Schnittstelle zur Verfügung steht.

5

8. Verfahren nach einem der vorstehenden Ansprüche,  
d a d u r c h g e k e n n z e i c h n e t,  
dass dem Anwender im Flow Chart-Editor, in Abhängigkeit von  
der zugrundeliegenden Maschinenprojektierung und/oder Hard-  
warekonfiguration adäquate Sprachmechanismen zur Verfügung  
gestellt werden.

10

9. Verfahren nach einem der vorstehenden Ansprüche,  
d a d u r c h g e k e n n z e i c h n e t,  
dass aus anwenderdefinierten Unterprogrammen (STUP) der tex-  
tuellen Sprache (ST) automatisch über einen Umsetzer (C) nach  
Art eines Compilers weitere grafische Elemente (E1-En) in  
Flow Chart-Notation (MCC) generiert werden, welche die Funk-  
tionsschnittstelle der entsprechenden Unterprogramme (STUP)  
enthalten und die dem Anwender (A1,A2) ebenfalls zur Verfü-  
gung gestellt werden.

20

10. Verfahren nach einem der vorstehenden Ansprüche,  
d a d u r c h g e k e n n z e i c h n e t,  
dass die automatisch generierten grafischen Elemente (E1-En)  
vom Anwender (A1,A2) als Sprachelemente des Flow Charts (MCC)  
verwendet werden.

25

11. Verfahren nach einem der vorstehenden Ansprüche,  
d a d u r c h g e k e n n z e i c h n e t,  
dass als textuelle Sprache (ST) Structured Text nach IEC 6-  
1131 verwendet wird.

30

12. Verfahren nach Anspruch 11,  
35 d a d u r c h g e k e n n z e i c h n e t,

dass ein Anwender zum Formulieren von Bedingungen beliebig zwischen den Darstellungsformen textuelle Sprache (ST), Kontaktplan (KOP) und/oder Funktionsplan (FUP) wechseln kann.

- 5 13. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) als Sprachelemente mindestens eine Schleife und/oder mindestens eine Parallelverzweigung vorhanden sind.
- 10 14. Verfahren nach Anspruch 13,  
dadurch gekennzeichnet,  
dass innerhalb der jeweiligen Parallelverzweigung die einzelnen Befehle im selben Interpolatortakt gestartet werden.
- 15 15. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) die Funktionsblöcke durch Maskeneingabe parametriert werden.
- 20 16. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) Funktionsblöcke zu Modulen zusammengefasst werden, die wiederum als Funktionsblöcke erscheinen.
- 25 17. Verfahren nach Anspruch 16,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) Module ineinander geschachtelt werden.
- 30 18. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) dem Anwender in den Funktionsblöcken für die Variablenzuweisung jeweils mehrere Zuweisungen möglich sind.

19. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass in der Flow Chart-Notation (MCC) die Funktionsblöcke,  
die Funktionen repräsentieren, die eine Zeitdauer beanspru-  
chen, Weiterschaltbedingungen enthalten.
20. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass die grafischen Elemente (El-En) des Flow Charts automa-  
tisch positioniert werden.
21. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass die grafischen Elemente (El-En) des Flow Charts automa-  
tisch miteinander verbunden werden.
22. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass das Flow Chart in der Anzeige verkleinert oder vergrö-  
ßert dargestellt wird.
23. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass durch Markierungen in der textuellen Sprache eine Rück-  
übersetzung in Flow Chart-Notation (MCC) möglich ist.
24. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass die Schritte a) bis c) in einem Sammelschritt ausgelöst  
werden.
25. Verfahren nach einem der vorstehenden Ansprüche,  
dadurch gekennzeichnet,  
dass während der Abarbeitung des Flow Chart Programms das je-  
weils aktuelle grafische Element (El-En) optisch auf der An-  
zeigeeinrichtung gekennzeichnet wird.

## Zusammenfassung

Verfahren zum Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungssteuerungen im Kontext der Flow

### 5 Chart Programmierung

Durch die Erfindung wird ein Verfahren zum Debuggen von Programmen für industrielle Steuerungen, insbesondere Bewegungs-

steuerungen im Kontext der Flow Chart Programmierung aufge-

10 zeigt, wobei den grafischen Elementen (E1-En) sog. Suspend-

Befehle zugeordnet werden. Durch den Einsatz eines Tasksteu-

ermechanismus im Run-Time-System (RTS) kann der Anwender (A1,

A2) auf Flow Chart-Ebene beim Debuggen einen Single-Step-

Modus und/oder einen Breakpoint-Modus verwenden.

15

FIG 2

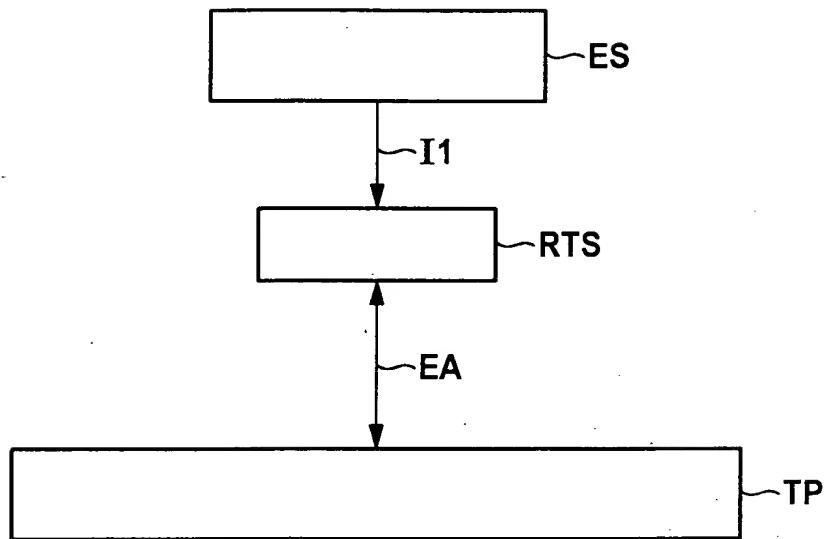


FIG 1

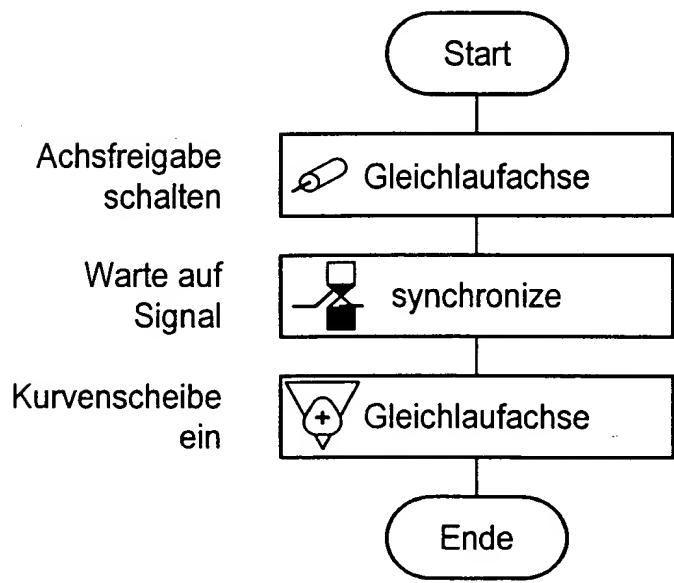


FIG 4

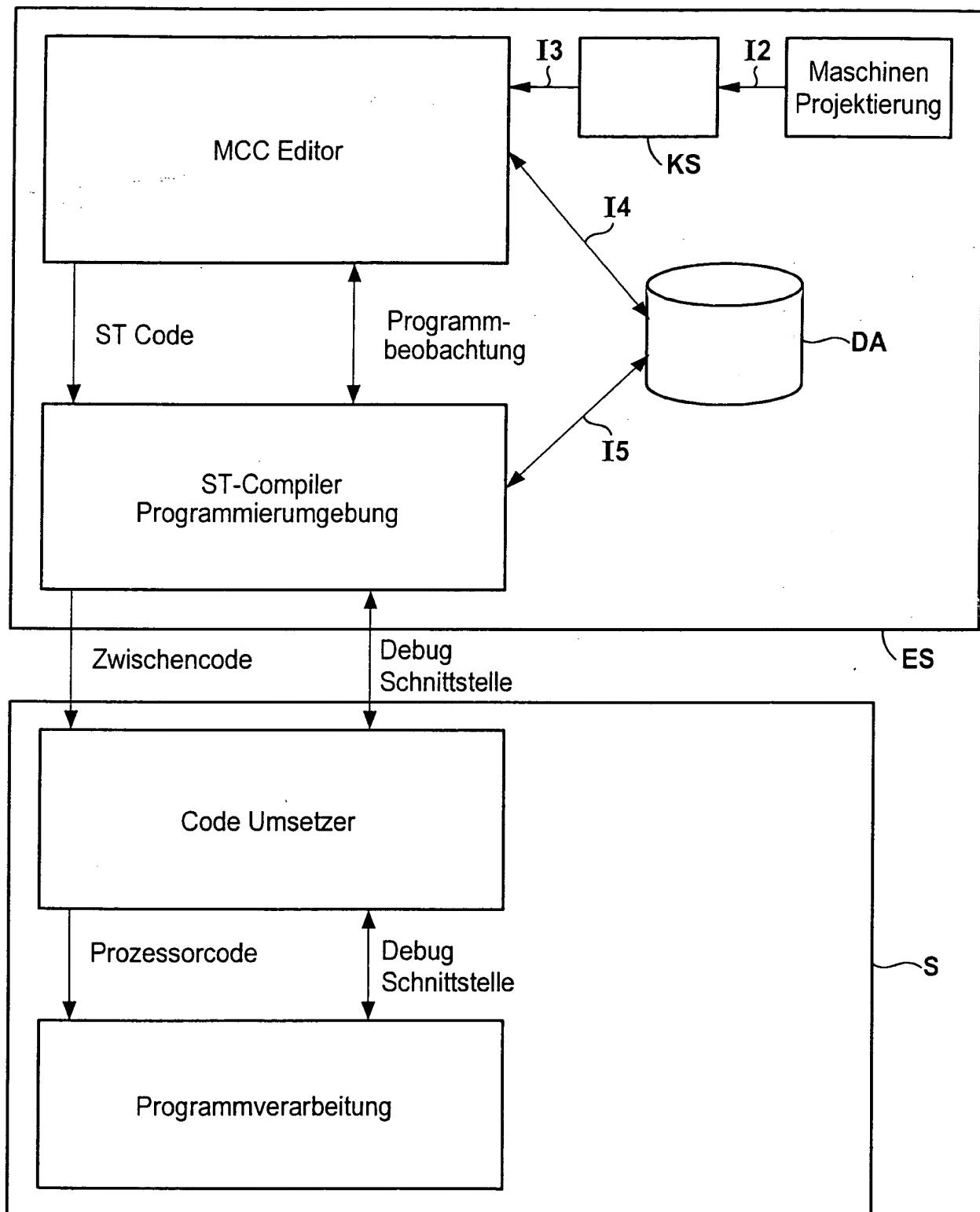


FIG 2

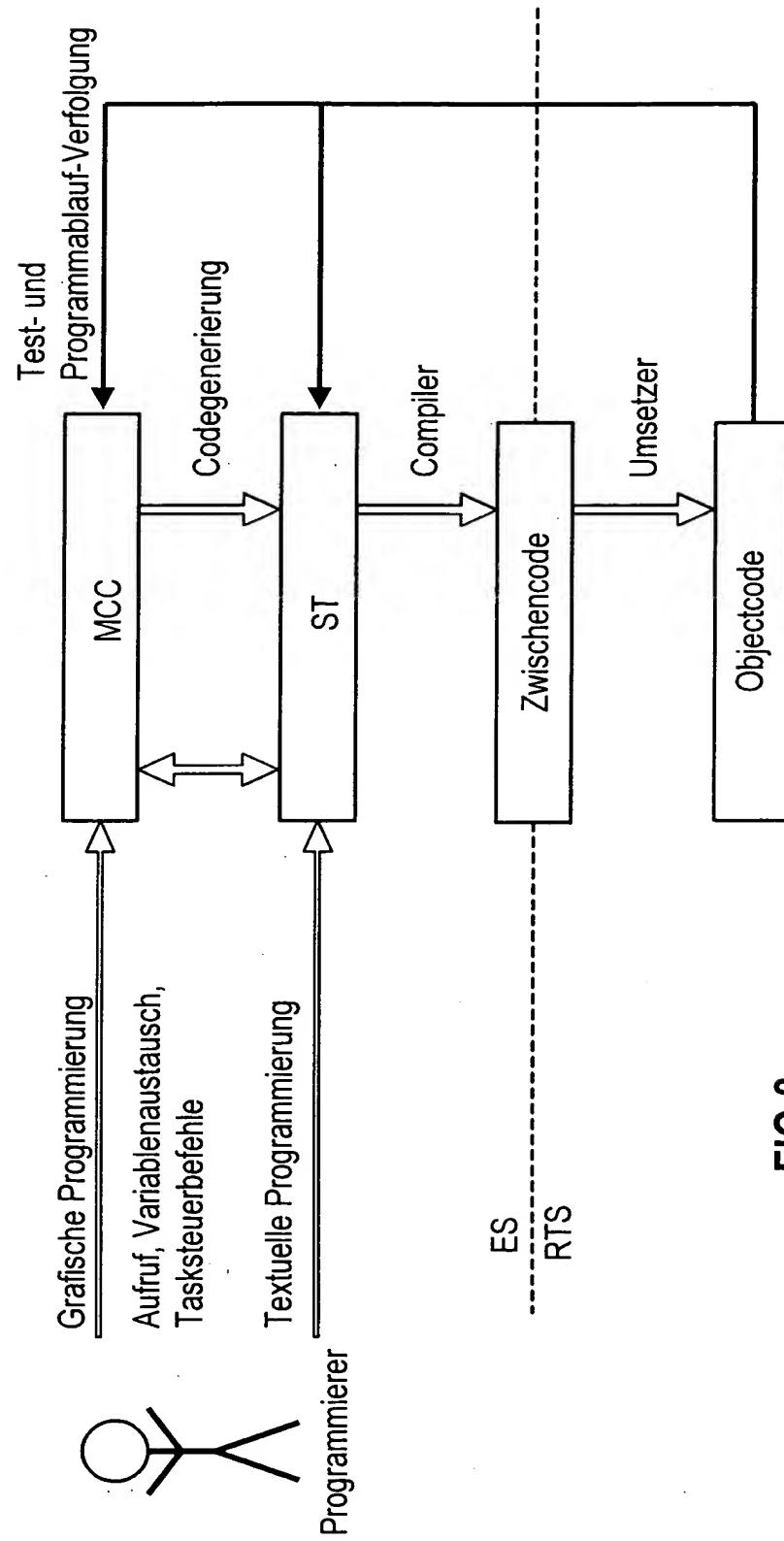


FIG 3

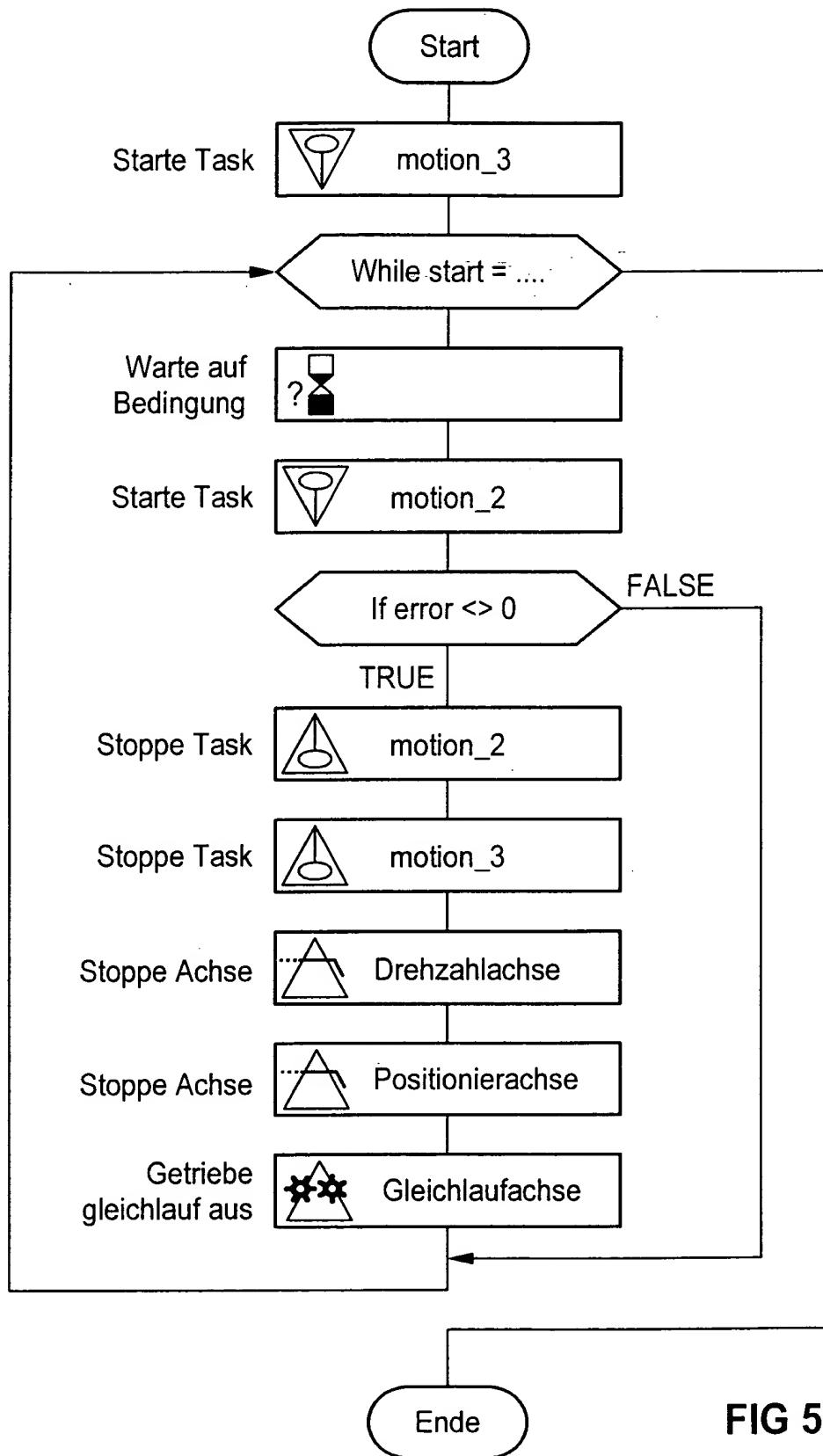


FIG 5

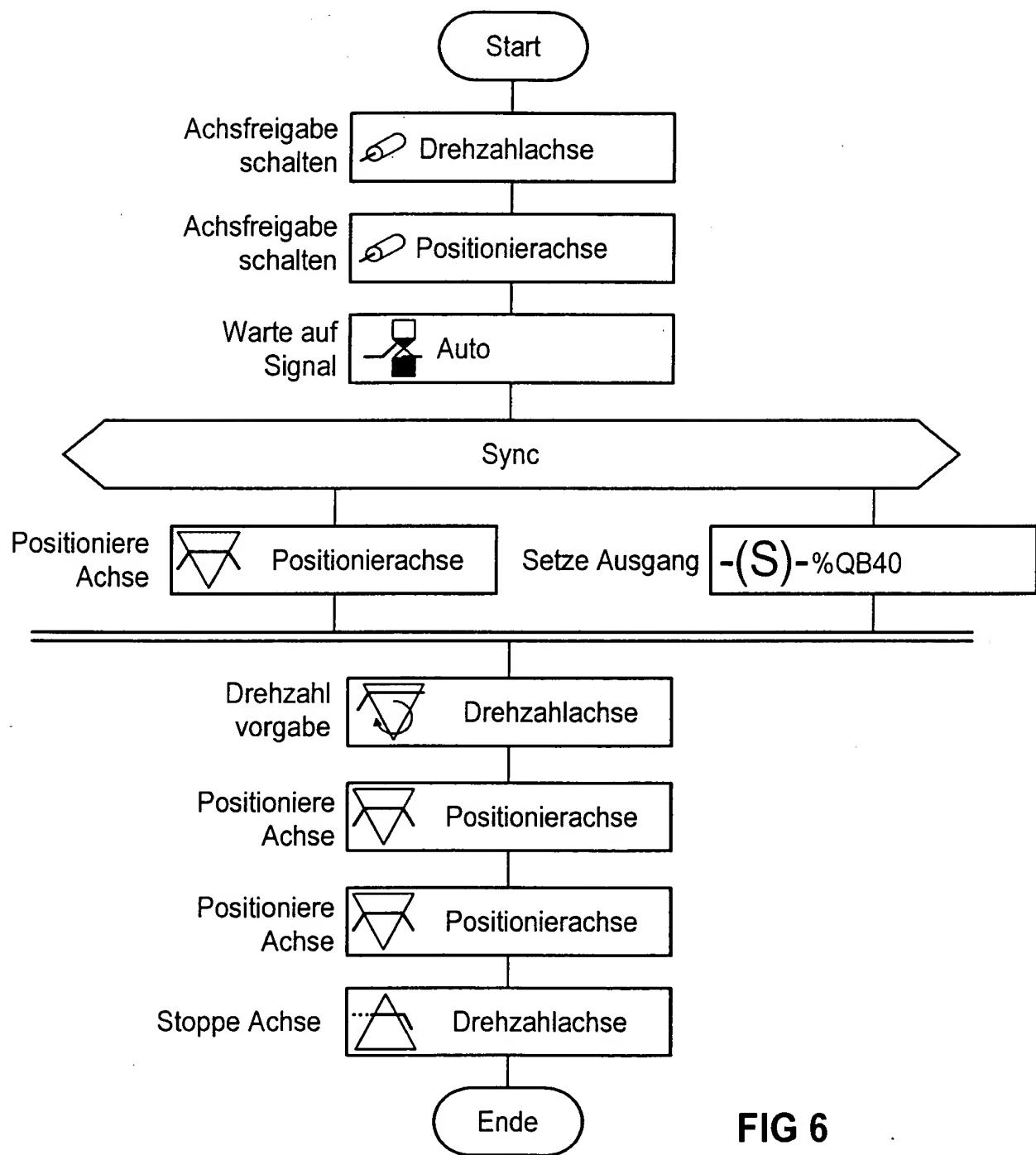
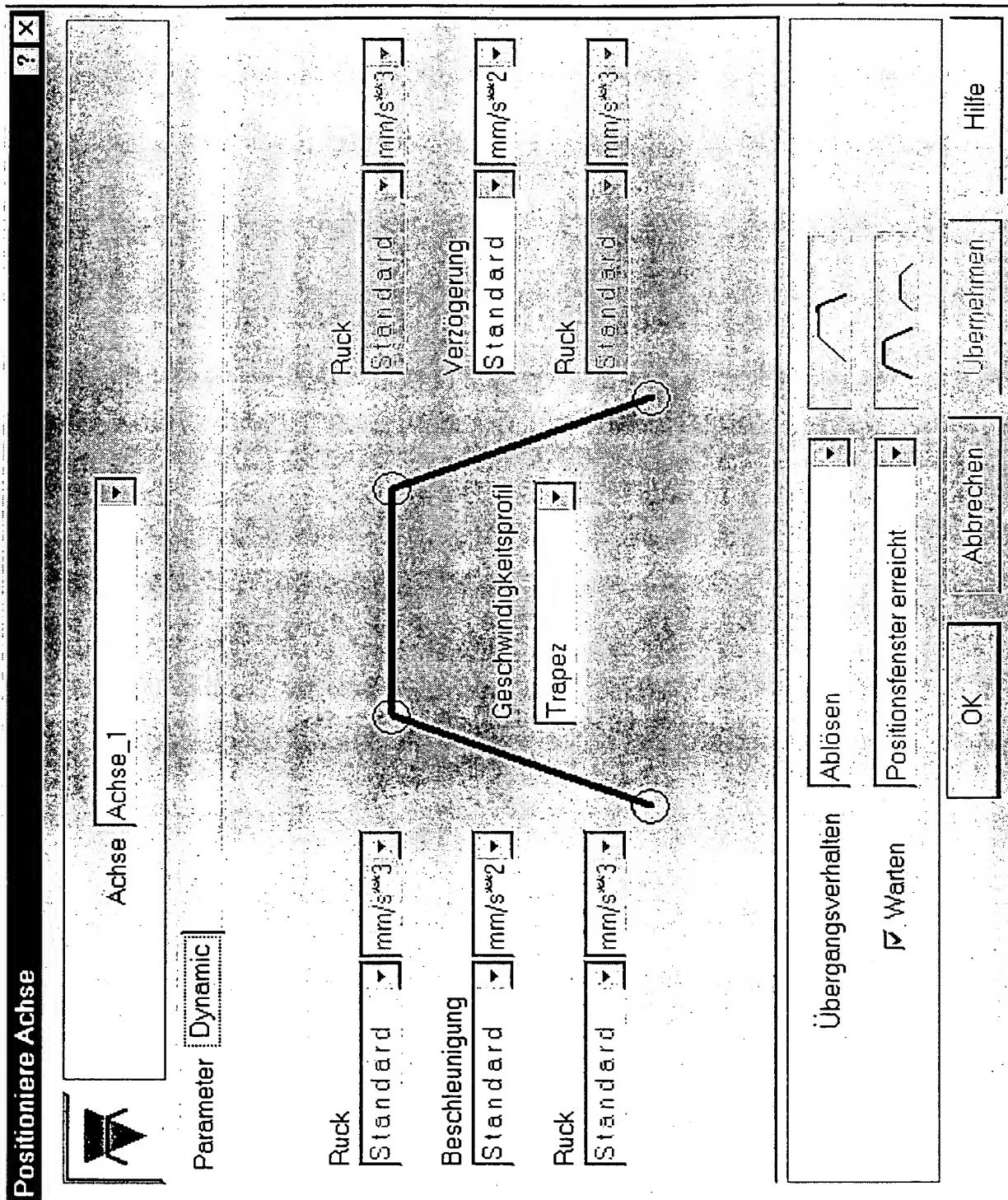


FIG 6

FIG 7



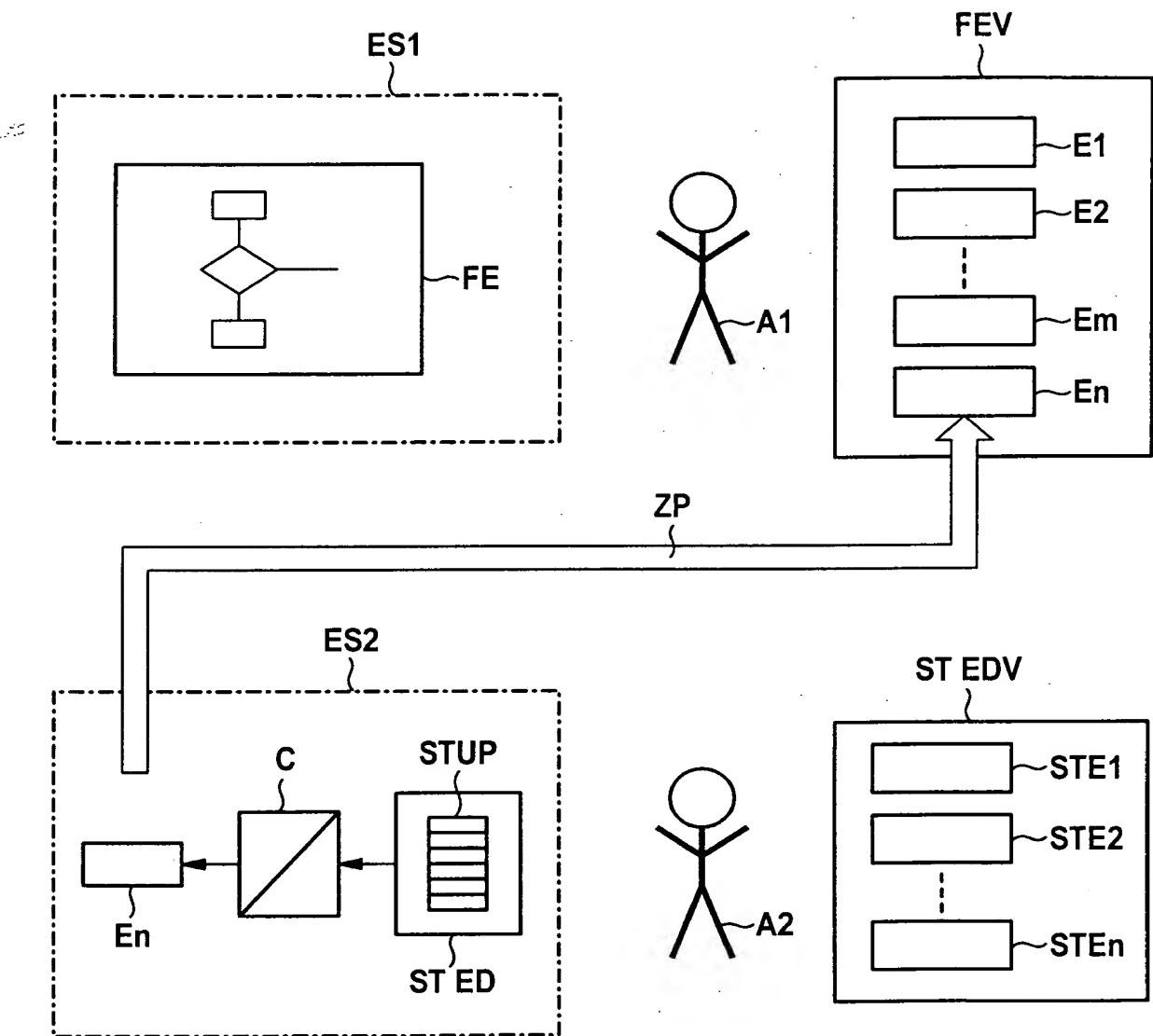


FIG 8

Drehzahl vorgabe		Aktiv. Getriebe- Gleichlauf / Gearing On		Stoppe Achse / Stop. Bewegung ist ein Befehl		Makro	
Drehmoment vorgabe		Schalte Gleichlauf aus		Stoppe Befehl		Unter- programm- aufruf	
Starte Achse lagegeregt		Aktiviere Kurvenscheibe / Canning		Setze Bewegung fort			
positioniere Achse		Schalte Kurvenscheibe aus					
		Parametriere Kurvenscheibe					

FIG 9